Theon Deployment

Tim Colles
Running specific Theon service elements themselves is trivial.

Its the local infrastructure integration that’s always going to be the bug bear, primarily authentication and authorization.

Summary
Running a devolved service as-is, treating Theon as an out of the box, shrink wrapped third party product.

No development framework, testing environment or change management.
A managed, recoverable, backed up and suitably configured instance of PostgreSQL with SSL.

Holds raw data, backup snapshots and audit logs.
• PostgreSQL 8.4, moving to 9.1
  – Oracle possible (SQL & PL/SQL are comparable)

• For service management we use LCFG:
  – lcfg-postgresql (configures, runs and does backups)
  – lcfg-rmirror (copies backups off-server and onto tape)
  – lcfg-cron (schedules backups)
  – lcfg-x509 (acquires certificates for SSL)

• Server spec:
  – RAID, 8GB (to hold entire database(s) plus sort/join queries)
  – Unlikely to see loading/scaling issues
  – Raw Data <1GB, Backup/Audit Data ~5GB
A managed, recoverable and suitably configured instance of Apache with SSL.

Hosts Portal web site. Holds no master data.
• Apache 2 with SSL

• For service management we use LCFG:
  – lcfg-apacheconf (configures and runs)
  – lcfg-x509 (acquires certificates for SSL)
  – lcfg-cron (for scheduled web page generation)

• TheonPortal
  – theon-portal, theon-portal-conduits: RPM or TGZ
  – gurgle: RPM or TGZ

• Server spec:
  – Unlikely to see loading/scaling issues
  – Document Root <100MB, TheonPortal <10MB
A managed, recoverable and suitably configured instance of Apache with SSL.

Hosts UI web application. Holds no master data.
• Apache 2 with SSL and WSGI
• For service management we use LCFG:
  – lcfg-apacheconf (configures and runs)
  – lcfg-x509 (acquires certificates for SSL)
• TheonUI
  – Python, Psycopg2
  – theon-ui, theon-ui-schema: RPM or TGZ
• Server spec:
  – Unlikely to see loading/scaling issues
  – UI Client/Server ~250MB, Schema ~35MB
Interface for all incoming data originating outside of Theon. Manages the feeds, synchronization and local override.

Hosted within the Database Service.
• Framework to accept and filter incoming feeds
• We run a local mail service using LCFG:
  – \texttt{lcfg-procmailrc} (configures filters, redirects to feed scripts)
  – \texttt{lcfg-postfix} (configures and runs mail service)
  – \texttt{lcfg-cron} (for scheduling data feeds which are pulled)
• TheonCoupler
  – \texttt{theon-coupler}, \texttt{theon-coupler-feeds}: RPM or TGZ
• Server spec:
  – Unlikely to see loading/scaling issues
  – Capacity to process/log incoming data, \(\sim\)1GB
Kerberos is the external authentication identity provider.

The Database uses Kerberos directly. The UI and Portal devolve authentication to an external Cosign service.
• Kerberized PostgreSQL
• Service Authentication:
  – lcfg-kerberos (obtains service ticket and maintains keytab)
• User Authentication:
  – Kerberos credentials
• Service Authentication:
  – Runs as service (does not use session user credentials)
  – lcfg-kerberos (obtains service ticket and maintains keytab)
  – lcfg-postgresql (on Database Service) maps portal service identity to a database account with read-only access role

• User Authentication:
  – Via Cosign, e.g. EASE
  – Apache configured with lcfg-cosign
  – Authentication used for location specific access control
  – Per-container access control generated from Theon
• User Authentication:
  - Does not run as a service, uses delegated user credentials
    • Connects “as” user, using native database permissions and audit trail
    • No direct access from UI Server to database without user credentials
  - Via Cosign with KCT Credential Delegation enabled
  - Apache configured with lcfg-cosign
  - EASE options:
    - Use delegated credentials if available, make available if not
    - Don’t, proxy via Informatics iFriend and per-user identity mappings
      • Pros: Easy; Cons: Extra web login for user
  - Use service account to “connect as user” from UI Server
    • Pros: Can use EASE; Cons: Security impact, needs small server mod
Mapping Kerberos identity to Database Service account. Associating accounts with data access roles.

The Database and the Portal directly use authorization. The UI devolves authorization to the Database Service.
• Functional Account Mapping, we use LCFG:
  – lcfg-postgresql (for functional accounts, portal and coupler and other downstream services)

• User Account Mapping and Role Mapping, we use LCFG:
  – lcfg-pgluser (configures and updates database)
  – Static data held in LCFG
  – Live data looked up in LDAP
  – LDAP fed from our Roles/Capabilities infrastructure
  – Roles/Capabilities data generated by Theon
  – Can pull live data from any source in principle, e.g. IDMS
  – Role mapping to database permissions is static in schema
Handling updates to Schema, Coupler, Portal and UI.
• Tools and Services
  - gurgle, theon-ui, theon-portal and theon-coupler
  - Update from new releases as and when

• Schema and Dependents:
  - theon-schema
  - theon-ui-schema
  - theon-portal-conduits
  - theon-coupler-feeds
  - Generally, but not always, will require planned downtime and careful (simultaneous) upgrade in combination with live Data Model alteration